ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ассистент |  |  |  | М. А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Обработка числовых последовательностей |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

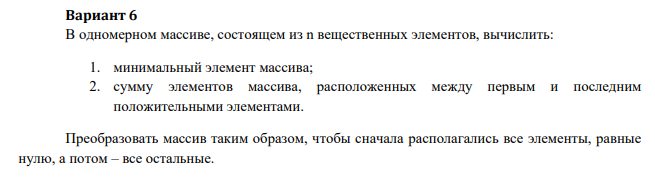
РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4135К |  |  |  |  |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**1.Цель работы:** Целью работы является изучение структуры данных одномерный массив.

**2.Задачи работы:** Задания на лабораторную работу приводятся в каждом варианте. При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаѐтся именованной константой. Примечание: массивы, созданные с помощью классов (таких, как например, array или vector из библиотеки STL) использовать запрещается. Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

**  
3.Описание функций:**  
main-основная функция  
**1)Имя: main**  
**Назначение:** создание динамического массива, запись в него чисел, выводит минимальный элемент массива, сумму элементов массива, которые располагаются между первым и последним положительными элементами

**Входные данные:** нет  
**Выходные данные:** нет

**Побочный эффект:** отсутствует.

**Тестовые данные:**   
**Ввод:** arr[8]{ -5,6,0,8,10,5,0,-5}

**Вывод:**

Минимальный элемент=-5

Сумма элементов массива=29

0 0 -5 6 8 10 5 -5   
**Прототип:** int main()  
**Псевдокод:**ввод размера массива с проверкой

создаём массив

вводим массив

выводим массив на экран

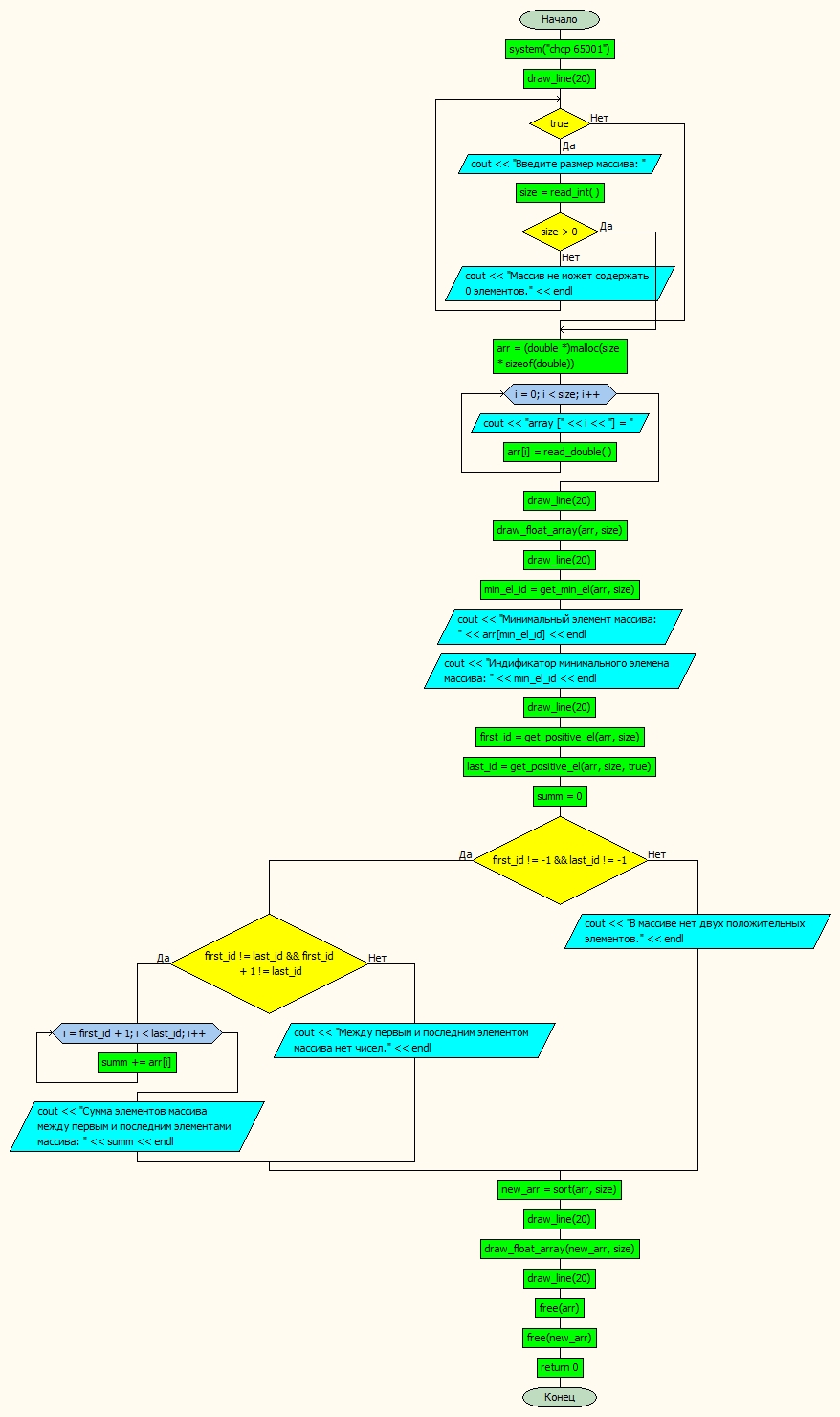
находим минимальный элемент массива

получаем идентификаторы первого и положительного элемента

условия для проверки (между числами должны находится ещё числа)

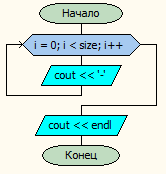
получаем сортированный массив

вывод массива

удаление массивов  
**Блок-схема:**  


**2)Имя: draw\_line()**  
**Назначение:** вывод линии в консоль   
**Входные данные:** size

**Выходные данные:** x  
**Побочный эффект:** отсутствует.   
**Прототип:** void draw\_line()  
**Псевдокод:**вывод символа ‘-’ size количество раз  
**Блок-схема:**

****

**3)Имя: read\_double()**  
**Назначение:** проверка корректности ввода(double)  
**Входные данные:** нет

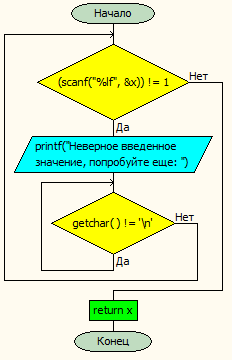
**Выходные данные:** x  
**Побочный эффект:** отсутствует.   
**Прототип:** double read\_double()  
**Псевдокод:**Ввод значения  
Если флаг ошибки, то вход в цикл

Пока не будет введен интер

Очистка буфера

Повтор ввода

**Блок-схема:**

****

**4)Имя: read\_int()**  
**Назначение:** проверка корректности ввода(int)  
**Входные данные:** нет

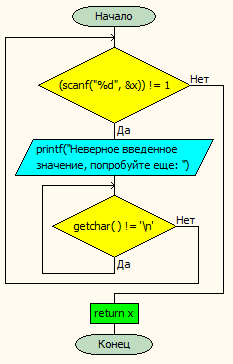
**Выходные данные:** x  
**Побочный эффект:** отсутствует.   
**Прототип:** int read\_int()  
**Псевдокод:**Ввод значения  
Если флаг ошибки, то вход в цикл

Пока не будет введен интер

Очистка буфера

Повтор ввода

**Блок-схема:**

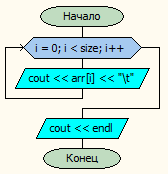
****

**5)Имя: draw\_float\_array( )**  
**Назначение:** вывод одномерного массива  
**Входные данные:** size,arr[]

**Выходные данные:** нет  
**Побочный эффект:** отсутствует.   
**Прототип:** void draw\_float\_array()  
**Псевдокод:**Вывод массива на экран пока элемент не станет равным размеру

И табуляция между элементами

**Блок-схема:**

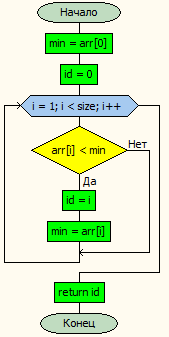
****

**6)Имя: get\_min\_el ()**  
**Назначение:** нахождение минимального элемента массива  
**Входные данные:** size,arr[]

**Выходные данные:** id  
**Побочный эффект:** отсутствует.   
**Прототип:** int get\_min\_el()  
**Псевдокод:**Перебираем массив пока не достигнем последнего элемента

если число меньше чем то что записано в min, то перезаписываем его

**Блок-схема:**

****

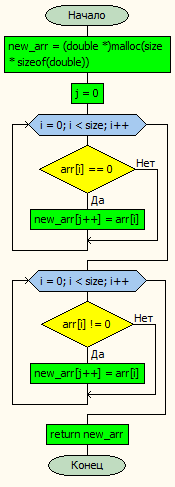
**7)Имя: sort ()**  
**Назначение:** сортировка массива(сначала 0, затем все остальные символы)  
**Входные данные:** size,arr[]

**Выходные данные:** new\_arr  
**Побочный эффект:** отсутствует.   
**Прототип:** double\* sort()  
**Псевдокод:**Создаем новый массив для записи отсортированного массива

Перебираем весь старый массив и если элемент равен 0, то записываем в новый

Перебираем весь старый массив и если элемент не равен 0, то записываем в новый

**Блок-схема:**

****

**8)Имя: get\_positive\_el ()**  
**Назначение:** получение идентификаторов первого и последнего положительного элемента в массиве

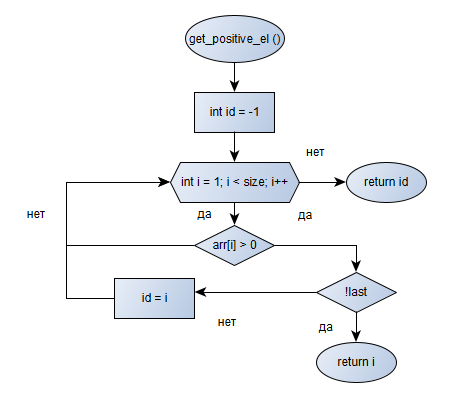
**Входные данные:** size,arr[],last

**Выходные данные:** id  
**Побочный эффект:** отсутствует.   
**Прототип:** int get\_positive\_el()  
**Псевдокод:**Цикл по всему массиву

если мы возвращаем первый элемент, то возвращаем индификатор здесь

если мы возвращаем последний элемент, то возвращаем индификатор здесь

**Блок-схема:**

****

**5.ЛИСТИНГ КОДА:**#include <iostream>

#include <cmath>

using namespace std;

// функция для нахождения минимального элемента (возвращает индификатор)

int get\_min\_el(double \*arr, int size) {

int min = arr[0];

int id = 0;

for (int i = 1; i < size; i++) {

// если число меньше чем то что записано в min, то перезаписываем его

if (arr[i] < min) {

id = i;

min = arr[i];

}

}

return id;

}

// функция для получения индификаторов первого или последнего положительного элемента

int get\_positive\_el(double \*arr, int size, bool last=false) {

int id = -1;

// цикл по всему массиву

for (int i = 1; i < size; i++) {

if (arr[i] > 0) {

// если мы возвращаем первый элемент, то возвращаем индификатор здесь

if (!last)

return i;

else

id = i;

}

}

// если мы возвращаем последний элемент, то возвращаем индификатор здесь

return id;

}

// функция для сортировки (сначала 0, потом всё остальное)

double \*sort(double \*arr, int size) {

// создаём новый массив

double \*new\_arr = (double\*) malloc(size \* sizeof(double));

int j = 0;

// сначала записываем все нули

for (int i = 0; i < size; i++) {

if (arr[i] == 0)

new\_arr[j++] = arr[i];

}

// потом записываем в массив всё остальное

for (int i = 0; i < size; i++) {

if (arr[i] != 0)

new\_arr[j++] = arr[i];

}

return new\_arr;

}

// функция для вывода линии в консоль (используется для разделения вывода)

void draw\_line(int size) {

for (int i = 0; i < size; i++)

cout << '-';

cout << endl;

}

// проверка ввода. (возвращает double)

double read\_double(){

double x;

while ( (scanf("%lf",&x) ) != 1 ) {

printf("Неверное введенное значение, попробуйте еще: ");

while(getchar() != '\n');

}

return x;

}

// проверка ввода. (возвращает int)

int read\_int(){

int x;

while ( (scanf("%d",&x) ) != 1 ) {

printf("Неверное введенное значение, попробуйте еще: ");

while(getchar() != '\n');

}

return x;

}

// функция для вывода одномерного массива

void draw\_float\_array(double \*arr, int size) {

for (int i = 0; i < size; i++)

cout << arr[i] << "\t";

cout << endl;

}

int main() {

// смена кодировки

system("chcp 65001");

// рисование линий в консоли (--------------------------)

draw\_line(20);

// ввод размера массива с проверкой

int size;

while (true) {

cout << "Введите размер массива: ";

size = read\_int();

if (size > 0) {

break;

} else {

cout << "Массив не может содержать 0 элементов." << endl;

}

}

// создаём массив

double \*arr = (double\*) malloc(size \* sizeof(double));

// вводим массив

for (int i = 0; i < size; i++){

cout << "array [" << i << "] = ";

arr[i] = read\_double();

}

draw\_line(20);

draw\_float\_array(arr, size); // выводим массив

draw\_line(20);

int min\_el\_id = get\_min\_el(arr, size);

cout << "Минимальный элемент массива: " << arr[min\_el\_id] << endl;

cout << "Индификатор минимального элемена массива: " << min\_el\_id << endl;

draw\_line(20);

// получаем индификаторы первого и полижительного элемента

int first\_id = get\_positive\_el(arr, size);

int last\_id = get\_positive\_el(arr, size, true);

int summ = 0;

// условия для проверка (между числами должны находится ещё числа)

if (first\_id != -1 && last\_id != -1) {

if (first\_id != last\_id && first\_id + 1 != last\_id) {

for (int i = first\_id + 1; i < last\_id; i++) {

summ += arr[i];

}

cout << "Сумма элементов массива между первым и последним элементами массива: " << summ << endl;

} else {

cout << "Между первым и последним элементом массива нет чисел." << endl;

}

} else {

cout << "В массиве нет двух положительных элементов." << endl;

}

// получаем сортированный массив

double \*new\_arr = sort(arr, size);

draw\_line(20);

draw\_float\_array(new\_arr, size); // выводим массив

draw\_line(20);

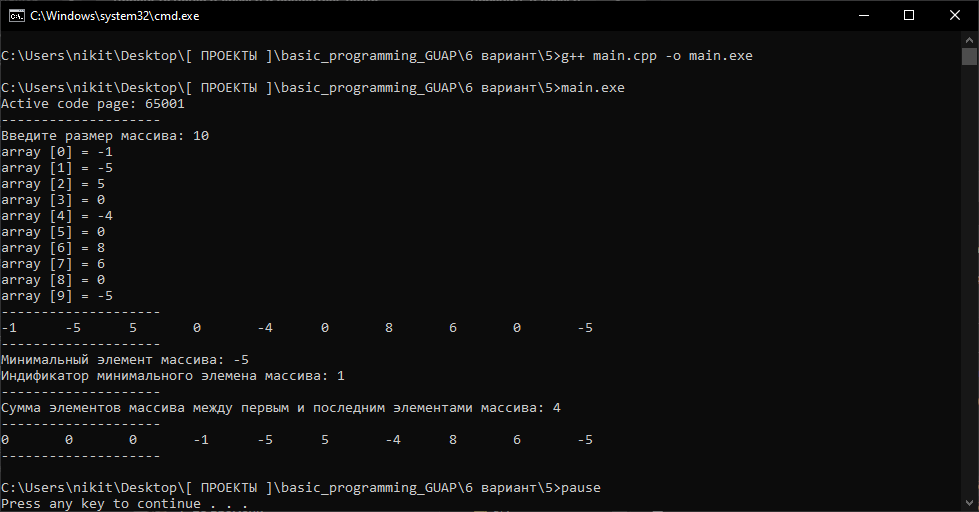
// удаление массивов

free(arr);

free(new\_arr);

return 0;

}

**6.Пример выполнения программы:**

**7.Анализ результатов и выводы:**

В ходе этой лабораторной работы мы научились создавать массивы, работать с содержимым массива, сортировать массив.

Из недостатков критических не обнаружено.

Из плюсов можно отметить, что программа работает корректно.